1. **Human Capital Management**
2. **AI HCM Product 1: AI-Powered Applicant Screening**

* 📛 Name Suggestion (Product-Level):

**SmartScreen HR**  
(Alternatives: HireMatch AI, TalentScore, CVIQ, FitRankHR)

* 🎯 Goal / Purpose:

To automate and enhance the candidate screening process by using AI to analyze CVs, match them with job descriptions (JDs), and provide HR professionals with fit scores and intelligent comparison dashboards to support better hiring decisions.

* 📦 Product Description:

**SmartScreen HR** is an AI-powered applicant screening tool that integrates with Applicant Tracking Systems (ATS) and HR platforms to streamline candidate evaluation. It uses NLP to parse resumes, extracts and standardizes candidate data, and matches it against job descriptions using machine learning. The system produces a match percentage, supports side-by-side applicant comparisons, and allows HR personnel to approve or reject candidates within a secure, intuitive dashboard. Evaluations are stored via secure APIs and can integrate into existing recruitment workflows.

* 🏗️ System Architecture:

Candidate Submission (via ATS / Email)

↓

NLP CV Parser & Skill Extractor

↓

AI Matching Engine (JD + Profile Match Scoring)

↓

Fit % Calculation + Applicant Comparison Module

↓

HR Dashboard (Review, Approve/Reject)

↓

Evaluation API (Secure Storage)

* 🖥️ UI Components (Frontend):

| **Component** | **Description** |
| --- | --- |
| <CandidateListPanel /> | List of parsed candidates and basic match score |
| <FitComparisonDashboard /> | Side-by-side comparison of applicants by % fit |
| <JDMatchViewer /> | View how candidate CV aligns with JD |
| <ApproveRejectControls /> | Buttons to shortlist, approve, or reject |
| <SkillGapHighlighter /> | Highlights mismatches or missing skills |
| <FilterSortControls /> | Sort/filter candidates by score, experience, education |
| <EvaluationHistoryPanel /> | Logs of past evaluations with notes |

* 🔐 Access Rights:

| **Role** | **Permissions** |
| --- | --- |
| HR Recruiter | Full access to candidate screening, filtering, and approval |
| Hiring Manager | Read-only access with commenting rights |
| Admin | Manage user roles, integrations, JD templates |
| Candidate | No access (interacts via ATS or email system only) |

* 🧠 AI Models and Recommendation Logic

📥 Input:

* CV/Resume (PDF, DOCX, etc.)
* Job Description (structured or text format)
* Skills dictionary (optional)
* HR feedback (for learning model fine-tuning)

📤 Output:

* Structured resume data (name, education, skills, experience, etc.)
* Fit score (e.g., 87%) based on JD-CV match
* Skill gap analysis
* Candidate comparison table
* Suggestion notes (optional)

🧠 AI Models Used:

* **NLP-based CV Parser:** Spacy or custom transformer-based model to extract resume sections
* **JD Matcher:** BERT or Sentence Transformers to compute semantic similarity between JD and CV
* **Fit Scoring Engine:** ML model to score candidate profile against JD using weighted features
* **Skill Extraction:** Named entity recognition and skill taxonomy lookup
* 🧰 Backend Architecture

📡 API Layer:

| **Endpoint** | **Description** |
| --- | --- |
| POST /cv/upload | Upload and parse resume |
| POST /job-description | Submit or select JD |
| GET /candidates/list | Return all matched candidates for a JD |
| GET /candidate/{id}/compare | Retrieve detailed comparison view |
| POST /candidate/{id}/decision | Save approve/reject decision |
| GET /evaluation/logs | View past decisions |

🔧 Services:

* **CV Parsing Service**: Extracts structured data from CVs
* **JD Matching Engine**: Computes fit score between CV and JD
* **Scoring & Recommendation Engine**: Assigns ranking
* **Secure Evaluation Store**: Persists decisions securely
* **ATS Integration Layer**: Connects to external HR systems (e.g., Greenhouse, Workday)
* 🛠️ Suggested Stack

| **Layer** | **Technology** |
| --- | --- |
| Frontend UI | React + Ant Design + D3.js for visuals |
| Backend API | Python (FastAPI) or Node.js |
| AI/NLP Models | HuggingFace Transformers (BERT), SpaCy, sklearn |
| Document Parsing | Apache Tika, pdfplumber |
| Secure API & Storage | OAuth2 + PostgreSQL + JWT |
| Integration | Webhooks & REST APIs for ATS/HRMS (e.g., BambooHR, SAP SuccessFactors) |
| Hosting | Docker + Kubernetes + AWS/GCP/Azure |

* 🔄 Data Flow Summary

1. Candidate resumes are received via email or ATS API

2. NLP parser extracts structured data (skills, experience, etc.)

3. HR uploads or selects the job description

4. JD and resume are semantically compared using AI model

5. Fit score is calculated and candidate is ranked accordingly

6. HR views candidates in dashboard, filters, and compares them

7. Approve/reject decisions are logged securely via API

8. All decisions can be audited or integrated back into ATS

1. **AI HCM Product 2: Skill Inference for Personalized Learning**

* 📛 Name Suggestion (Product-Level):

**SkillLens AI**  
(Alternatives: TalentMap, LearnPath AI, SkillRadar, InsightLearn)

* 🎯 Goal / Purpose:

To automatically infer latent (unlisted or undeclared) skills of employees based on their resumes, work history, and feedback data—then recommend personalized learning paths directly within their profiles to support career development.

* 📦 Product Description:

**SkillLens AI** analyzes employee profiles using machine learning to uncover potential or hidden skills. It enriches the employee profile with a "Potential Skills" tab and provides personalized course recommendations based on the inferred skills gap. Employees can confirm or reject the inferred skills, feeding back into the model. The system tags all skills using a centralized taxonomy and integrates seamlessly with the organization’s internal Learning Management System (LMS) to offer contextual training options.

* 🏗️ System Architecture:

Employee Profile Data (CV, JD, Work History, Feedback)

↓

Skill Inference Engine (ML)

↓

Potential Skills Generation + Confidence Score

↓

Skill Tagger (Taxonomy-Aligned)

↓

Learning Path Recommendation

↓

UI (Employee Dashboard with Confirm/Reject + Training Suggestions)

↓

LMS Integration + Feedback Loop

* 🖥️ UI Components (Frontend):

| **Component** | **Description** |
| --- | --- |
| <SkillProfileTab /> | New tab in profile showing current + potential skills |
| <SkillCard /> | Displays inferred skill with confidence % |
| <ConfirmRejectSkill /> | Employee confirms/rejects inferred skill |
| <RecommendedCoursesPanel /> | Course suggestions mapped to skill gaps |
| <LearningHistoryTimeline /> | Tracks course completion status |
| <TaxonomyTagViewer /> | Shows linked taxonomy and skill hierarchy |
| <FeedbackToModelWidget /> | Optional feedback tool for improving inferences |

* 🔐 Access Rights:

| **Role** | **Permissions** |
| --- | --- |
| Employee | View/edit skill profile, confirm skills, access recommended courses |
| HR/L&D Manager | View skills analytics, monitor upskilling, manage taxonomy |
| Admin | Manage AI settings, LMS integration, taxonomy imports |
| LMS System | Pull recommended courses and feed completion data |

* 🧠 AI Models and Recommendation Logic

📥 Input:

* Employee CV or resume
* Internal job history
* Feedback records or performance reviews
* Known skill tags (declared)
* Learning history (courses completed)

📤 Output:

* Inferred skill list with confidence scores (e.g., “SQL – 88%”)
* Personalized learning plan (LMS-linked)
* Skill tag (taxonomy-mapped)
* Confirmation request to employee

🧠 AI Models Used:

* **Skill Inference Engine:** ML model (XGBoost or neural net) trained on profile-to-skill mapping
* **NLP Embedding + Similarity Analysis:** For identifying implicit skill relationships
* **Taxonomy Tagger:** Matches inferred skills to standard frameworks (e.g., ESCO, SFIA)
* **Recommender System:** Suggests courses using collaborative filtering or content-based filtering
* 🧰 Backend Architecture

📡 API Layer:

| **Endpoint** | **Description** |
| --- | --- |
| GET /employee/{id}/skills | Retrieve known + inferred skills |
| POST /employee/{id}/skills/confirm | Confirm or reject inferred skill |
| GET /employee/{id}/recommendations | Return course list for missing skills |
| GET /taxonomy/skills | Fetch taxonomy-linked skill hierarchy |
| POST /feedback | Employee or HR feedback for inference engine |
| POST /integration/lms/push | Send recommended courses to LMS |

🔧 Services:

* **Inference Service:** Skill deduction from multiple employee data sources
* **Taxonomy Mapping Service:** Tags skills against master skill database
* **Recommendation Engine:** Maps skill gaps to learning items
* **Profile Enrichment Service:** Updates employee profile with dynamic content
* **LMS Sync Module:** Sends learning recommendations and syncs progress
* 🛠️ Suggested Stack

| **Layer** | **Technology** |
| --- | --- |
| Frontend UI | React + Tailwind CSS + Chart.js |
| Backend API | Python (FastAPI or Flask) |
| ML Models | Scikit-learn, TensorFlow, spaCy |
| NLP & Embeddings | BERT, Sentence Transformers |
| Database | PostgreSQL + ElasticSearch for search |
| Taxonomy DB | JSON-based or relational taxonomy tables |
| Integration Layer | RESTful APIs or LMS connectors (e.g., Moodle, Cornerstone) |
| Hosting | Docker + AWS ECS or Azure App Service |

* 🔄 Data Flow Summary

1. Employee uploads CV or data auto-ingested from HRIS/ATS

2. Inference Engine processes input and deduces potential skills

3. Skills are tagged to internal taxonomy

4. UI displays inferred skills with confirm/reject options

5. System recommends learning paths via LMS

6. Employee feedback (accept/reject) improves model over time

7. All skills and learning progress are logged in central skill profile

1. **AI HCM Product 3: 360 Report Summarization & Insight Generation**

* 📛 Name Suggestion (Product-Level):

**360Insights AI**  
(Alternatives: FeedbackLens, Clarity360, PulseAI, Reflect360)

* 🎯 Goal / Purpose:

To automate the summarization of 360-degree feedback reports using generative AI, highlight trends across peer/manager/self inputs, and suggest next-step actions for personal development, all while ensuring secure, role-based data access.

* 📦 Product Description:

**360Insights AI** connects to multiple feedback sources via API, aggregates 360-degree feedback from managers, peers, and self-assessments, and then uses LLMs (Large Language Models) to generate concise summaries. It visualizes trends over time and segments feedback sources clearly. The system also recommends growth actions based on strengths and weaknesses identified in the report. Access to reports is permission-based, maintaining confidentiality.

* 🏗️ System Architecture:

Multiple Feedback Sources (e.g., HRMS, Survey Tools, Email)

↓

Feedback Aggregator API

↓

LLM Summarizer (Group-wise + Overall)

↓

Trend Analyzer + Action Recommendation Engine

↓

Role-based View UI (Manager, HR, Employee)

↓

Secure Storage + Audit Logs

* 🖥️ UI Components (Frontend):

| **Component** | **Description** |
| --- | --- |
| <SummaryDashboard /> | Overall summary with key themes |
| <TrendVisualizationChart /> | Feedback trends over time by category |
| <FeedbackSourceTabs /> | View by peer, manager, self, etc. |
| <AIActionSuggestions /> | Personalized next-step guidance |
| <RoleBasedViewPanel /> | Adjusted views per user role |
| <RawFeedbackViewer /> | (Optional) View original verbatim comments |
| <DownloadOptions /> | Export report in PDF/Excel |

* 🔐 Access Rights:

| **Role** | **Permissions** |
| --- | --- |
| Employee | View own summarized report and suggestions |
| Manager | View direct report summaries, peer-level insight (if allowed) |
| HR/L&D | View all reports, export insights, monitor trends |
| Admin | Configure access rules, integrate feedback sources |

* 🧠 AI Models and Recommendation Logic

📥 Input:

* Raw 360-degree feedback (textual comments + ratings)
* Feedback source metadata (manager, peer, self, etc.)
* Time-series data for trend detection

📤 Output:

* Summarized report by category and source
* Visualization of changes over time
* AI-generated next-step recommendations (e.g., improve delegation skills)
* Highlighted strengths and focus areas

🧠 AI Models Used:

* **LLM Summarizer (e.g., GPT-based):** Generates concise summaries from feedback
* **Sentiment Analysis & Topic Modeling:** Extracts major themes and tones
* **Trend Analyzer:** Time-based comparison model for behavioral changes
* **Action Generator:** Template-based prompt engine with behavioral insights (e.g., “Based on recurring feedback about communication clarity…”)
* 🧰 Backend Architecture

📡 API Layer:

| **Endpoint** | **Description** |
| --- | --- |
| POST /feedback/ingest | Upload raw feedback data |
| GET /report/{userId} | Retrieve full summary report |
| GET /trends/{userId} | Show changes over time |
| POST /generate-actions/{userId} | Get AI-generated suggestions |
| GET /roles/view-permissions | Enforce access control per user role |
| GET /export/{userId}?format=pdf | Export report |

🔧 Services:

* **Data Aggregator:** Collects structured and unstructured feedback data
* **Summarization Engine:** Orchestrates LLM calls and stores outputs
* **Trend Analytics Module:** Tracks behavioral changes over multiple cycles
* **Action Recommendation Service:** Maps behavior to growth suggestions
* **Access Control Layer:** Manages what each role can view or download
* 🛠️ Suggested Stack

| **Layer** | **Technology** |
| --- | --- |
| Frontend UI | React + D3.js/Chart.js for data visualization |
| Backend API | Python (FastAPI or Flask) |
| LLM Integration | OpenAI GPT, Azure OpenAI, or local LLaMA via LangChain |
| Sentiment Analysis | spaCy + VADER + custom classifier |
| Database | PostgreSQL for structured data, MongoDB for feedback text |
| Auth & Access | OAuth2 / RBAC system |
| Hosting | Kubernetes on GCP/AWS, S3 for report storage |

* 🔄 Data Flow Summary

1. Feedback data ingested via API or HR system connection

2. Feedback is tagged and categorized (peer, manager, self)

3. LLM processes and summarizes grouped feedback

4. Trends over time visualized and compared to past reports

5. System generates actionable development advice

6. Employee views secure, personalized dashboard

7. HR/Manager accesses based on permissions, exports reports if needed

1. **AI HCM Product 4: JD & Interview Enhancement**

* 📛 Name Suggestion (Product-Level):

**SmartHire Composer**  
(Alternatives: JDGenie, RoleCraft AI, IntelliJD, AskHire)

* 🎯 Goal / Purpose:

To assist HR teams and hiring managers in drafting high-quality, standardized job descriptions (JDs) and generating relevant interview questions using AI, while ensuring alignment with internal role libraries and job family taxonomies.

* 📦 Product Description:

**SmartHire Composer** is an AI-powered assistant that enhances the creation and management of job descriptions and interview questions. It leverages GenAI to recommend job content, responsibilities, required skills, and role-specific interview questions. Users can interactively edit and preview the JD with real-time AI suggestions. All versions are stored with full edit history. The system uses an internal job family taxonomy and sample JDs for personalization and consistency.

* 🏗️ System Architecture:

Internal JD/Role Library + Job Family Taxonomy

↓

Prompt Generator Engine (JD + Interview Context)

↓

LLM API (JD Generation & Question Generation)

↓

JD Editor Interface + Versioning API

↓

Role-Based Preview & Export

* 🖥️ UI Components (Frontend):

| **Component** | **Description** |
| --- | --- |
| <JDComposerEditor /> | WYSIWYG-style JD editor with live suggestions |
| <RoleSelector /> | Choose or browse job roles by department/family |
| <AIPromptsSidebar /> | Suggested duties, skills, qualifications, etc. |
| <InterviewQuestionGenerator /> | View or regenerate interview questions |
| <PreviewPanel /> | Live preview with formatting and export options |
| <JDVersionHistory /> | View/edit historical versions with timestamps |
| <ExportOptions /> | Export to Word/PDF, sync with ATS |

* 🔐 Access Rights:

| **Role** | **Permissions** |
| --- | --- |
| HR/Recruiter | Create, edit, approve JDs, generate interview questions |
| Hiring Manager | Request JD drafts, propose edits, review suggestions |
| Admin | Manage job family taxonomy, role templates, permissions |
| Viewer (Team) | View finalized JDs, export only (no edits) |

* 🧠 AI Models and Recommendation Logic

📥 Input:

* Role title, department, seniority level
* Selected job family category
* Existing JD data (if editing)
* Internal sample JDs and question library

📤 Output:

* AI-generated job description content (summary, responsibilities, qualifications)
* Interview questions based on role requirements
* Optional prompts: soft skills, behavioral assessments
* Versioned JD document with edit traceability

🧠 AI Models Used:

* **Prompt-tuned LLM (e.g., GPT-4, Claude, or fine-tuned LLaMA):**  
  Generates JD text and interview questions using contextual prompts
* **Job Role Classifier (internal model):**  
  Maps entered role to taxonomy for better prompt enrichment
* **Similarity Engine (cosine vector matcher):**  
  Suggests close matches to internal JD templates
* 🧰 Backend Architecture

📡 API Layer:

| **Endpoint** | **Description** |
| --- | --- |
| POST /jd/generate | Generate JD from title + role |
| POST /interview/generate | Generate interview questions |
| GET /jd/version-history/{id} | Retrieve past JD versions |
| PUT /jd/update | Update/edit current JD |
| GET /roles/list | List available roles and taxonomy |
| `GET /jd/export/{id}?format=pdf | docx` |

🔧 Services:

* **LLM Prompt Orchestrator:** Constructs enriched prompts for JD generation
* **JD Versioning Service:** Tracks edit history and rollback capability
* **Role-Taxonomy Mapper:** Normalizes job roles into predefined families
* **Question Generator:** Generates situational/technical/behavioral questions
* **Access Control Service:** Manages edit/view permissions
* 🛠️ Suggested Stack

| **Layer** | **Technology** |
| --- | --- |
| Frontend UI | React, Draft.js/Slate for JD editing, Tailwind CSS |
| Backend API | Node.js or Python (FastAPI) |
| LLM Integration | OpenAI, Azure OpenAI, or private LLM endpoint |
| Data Store | PostgreSQL (JD versions), Redis (session/store) |
| Storage | S3/GCS for JD files |
| Auth & Permissions | OAuth2 + RBAC + JWT |
| Deployment | Dockerized services on Kubernetes or ECS |

* 🔄 Data Flow Summary

1. User selects or enters job title and role context

2. System references internal taxonomy + JD library

3. Prompt engine enriches input and calls LLM

4. JD content and interview questions are returned

5. User reviews, edits, previews JD via WYSIWYG interface

6. Changes are versioned and stored with metadata

7. Finalized JD can be exported or synced with recruitment system

1. **AI HCM Product 5: Career Path & Succession Planning**

* 📛 Name Suggestion (Product-Level):

**PathwayAI**  
(Alternatives: CareerNext, TalentNavigator, SuccessionAI, FutureFit)

* 🎯 Goal / Purpose:

To enable organizations to visualize career pathways for employees, recommend next potential roles based on their profile and readiness, and assist HRBPs and managers in identifying successors and planning leadership pipelines.

* 📦 Product Description:

**PathwayAI** is an AI-driven solution that helps organizations with career development and succession planning. It visualizes employee career trajectories, recommends future-fit roles, scores their readiness based on performance and competency data, and supports HRBPs with dashboards and reports to manage succession pipelines. The system uses behavioral data, job transitions, and internal competency models to infer and predict optimal career paths.

* 🏗️ System Architecture:

Employee Profile + Competency & Performance Data

↓

Career Graph & Role Transition Models

↓

AI Readiness Scoring Engine

↓

Succession Dashboard + Career Path UI

↓

Report Export + Manager/HRBP Tools

* 🖥️ UI Components (Frontend):

| **Component** | **Description** |
| --- | --- |
| <CareerPathVisualizer /> | Interactive timeline showing employee's current position and recommended next roles |
| <RoleRecommendationPanel /> | AI-suggested future roles with fit score and rationale |
| <SuccessionDashboard /> | Overview of key roles, readiness pool, risk indicators |
| <ReadinessScoreWidget /> | Visual representation of employee readiness |
| <ExportReportModal /> | Export PDF/Excel reports for HRBP or line managers |
| <CompetencyGapViewer /> | Highlight gaps between current skills and target roles |
| <FilterPanel /> | Filter by department, level, criticality |

* 🔐 Access Rights:

| **Role** | **Permissions** |
| --- | --- |
| Employee | View career path, confirm interest, request mentoring/training |
| Manager | View team career paths, succession candidates, readiness scores |
| HRBP | Access succession dashboard, download reports, assign development plans |
| Admin | Manage role graph, competency mappings, access rules |

* 🧠 AI Models and Recommendation Logic

📥 **Input:**

* Current role, experience, historical transitions
* Performance ratings, competency scores, behavioral traits
* Internal role taxonomy and transition graph
* Learning history and certifications

📤 Output:

* Recommended next roles with scoring (fit + readiness)
* Timeframe estimation to reach readiness
* Succession candidate ranking for key roles
* Development plan recommendations to close gaps

🧠 AI Models Used:

* **Career Graph Inference Model:** Learns transition probabilities between roles based on historical data
* **Readiness Scoring Model:** Combines competency levels, performance trends, and tenure to compute role readiness
* **Behavioral Fit Predictor:** Maps personality/feedback data to role success patterns
* **Succession Risk Analyzer:** Identifies risk areas where successor bench strength is low
* 🧰 Backend Architecture

📡 API Layer:

| **Endpoint** | **Description** |
| --- | --- |
| GET /career-path/{employeeId} | Fetch career suggestions for an employee |
| POST /succession-dashboard/filter | Get filtered succession planning data |
| GET /readiness-score/{employeeId} | Compute and return readiness metrics |
| POST /export-report | Export dashboard insights as PDF/Excel |
| GET /role-transitions | List of valid transitions from a role |
| POST /admin/role-graph-update | Update career path mappings |

🔧 Services:

* **Career Graph Engine:** Stores and computes valid transitions and probabilities
* **Succession Planning Service:** Renders dashboards and tracks high-potential successors
* **Readiness Scoring Engine:** Aggregates inputs and applies business rules
* **Access Control & Data Masking:** Ensures proper access to sensitive career/performance data
* 🛠️ Suggested Stack

| **Layer** | **Technology** |
| --- | --- |
| Frontend UI | React + D3.js (for path visualization), Tailwind CSS |
| Backend API | Node.js / FastAPI (Python) |
| AI/ML Pipeline | Scikit-learn, XGBoost for scoring; Neo4j for career graph |
| Data Store | PostgreSQL (profile data), Redis (cache), S3 for exports |
| Authentication | OAuth2 + Role-Based Access (RBAC) |
| Deployment | Kubernetes or ECS with CI/CD pipeline |

* 🔄 Data Flow Summary

1. System pulls employee profile, skills, and performance history

2. Role transition graph and internal job taxonomy define valid career paths

3. AI models score fit and readiness for future roles

4. UI displays career path, next roles, and supporting recommendations

5. HRBPs and managers access succession dashboards and export insights

6. All actions and changes are tracked with secure audit logs

1. **AI HCM Product 6: Skills Taxonomy & Architecture Creation**

* 📛 Name Suggestion (Product-Level):

**SkillStruct**  
(Alternatives: TaxoBuilder, SkillArchitect, CompetencyMapAI, ClusterCore)

* 🎯 Goal / Purpose:

To enable organizations to design, maintain, and evolve a comprehensive skills taxonomy through an AI-assisted platform that can automatically group and structure skills based on job descriptions (JDs) and CVs, supporting scalable talent frameworks.

* 📦 Product Description:

**SkillStruct** is an AI-powered system that helps HR teams and competency architects manage and evolve organizational skills frameworks. It provides a drag-and-drop interface to structure skills by group/function, supports importing/exporting taxonomies (CSV/JSON), and leverages machine learning to auto-cluster skills and detect new ones from sources like CVs and job descriptions. The system also includes version control to track changes and updates across time.

* 🏗️ System Architecture:

Job Descriptions / CV Data

↓

AI Skill Extraction + Clustering

↓

Skill Taxonomy Builder UI

↓

Version Management & Export System

↓

Skill JSON / CSV Repository

* 🖥️ UI Components (Frontend):

| **Component** | **Description** |
| --- | --- |
| <TaxonomyTreeBuilder /> | Drag-and-drop interface for managing skill hierarchy |
| <SkillClusterSuggestionPanel /> | Displays AI-recommended skill groups and relationships |
| <VersionHistoryViewer /> | Shows past versions with diff and rollback |
| <ImportExportPanel /> | Upload/download taxonomy in JSON or CSV |
| <SkillDetailModal /> | Edit skill attributes like aliases, definitions, tags |
| <SearchAndTagWidget /> | Search, assign skills to groups, apply business tags |

* 🔐 Access Rights:

| **Role** | **Permissions** |
| --- | --- |
| Admin | Full control over taxonomy editing, imports/exports, versioning |
| HR Architect | Create/edit skills, use clustering suggestions, manage structure |
| Manager | View-only access to taxonomy structure |
| Viewer | Read-only access, no editing rights |

* 🧠 AI Models and Recommendation Logic

📥 **Input:**

* Internal job descriptions (past & current)
* Employee CV/resume datasets
* Manually entered or imported skill lists

📤 Output:

* Grouped skill clusters
* Suggested skill hierarchies (taxonomy trees)
* Synonym detection for similar skills
* New or missing skill recommendations

🧠 AI Models Used:

* **Skill Embedding & Clustering Model:** Uses transformer embeddings (e.g., SBERT) to vectorize and group similar skill terms
* **NER + Keyword Extraction Model:** Identifies and normalizes skill-related terms in unstructured text
* **Synonym Mapping Engine:** Recognizes and merges skill variants (e.g., “Excel” vs. “Microsoft Excel”)
* **Drift Detection Logic:** Flags outdated or overly granular skills for review
* 🧰 Backend Architecture

📡 API Layer:

| **Endpoint** | **Description** |
| --- | --- |
| POST /extract-skills | Extracts and normalizes skills from uploaded JD/CV |
| POST /cluster-skills | Returns clustered groups based on similarity |
| GET /taxonomy | Fetches current or specific version of taxonomy |
| POST /taxonomy/update | Save new taxonomy structure |
| POST /taxonomy/version/rollback | Revert to a previous taxonomy version |
| POST /taxonomy/export | Export taxonomy to CSV/JSON |
| POST /taxonomy/import | Import and parse taxonomy from file |

🔧 Services:

* **Skill Extraction Service**
* **Clustering & Embedding Service**
* **Version Control Service**
* **Export/Import Service**
* **Data Validation & Cleanup Pipeline**
* 🛠️ Suggested Stack

| **Layer** | **Technology** |
| --- | --- |
| Frontend UI | React + React Flow for drag-drop trees, Tailwind CSS |
| Backend API | FastAPI (Python) |
| AI/ML Models | SpaCy + SBERT + HDBSCAN for clustering |
| Data Store | MongoDB (for flexible taxonomy structure), PostgreSQL (metadata), Redis (caching) |
| File Storage | Amazon S3 or Azure Blob Storage |
| Authentication | OAuth2 + Role-Based Access |
| Deployment | Docker + Kubernetes with CI/CD pipelines |

* 🔄 Data Flow Summary

1. Admin uploads JD or CV data

2. AI model extracts skill terms and clusters similar ones

3. Suggested clusters appear in the UI for review or direct adoption

4. Admin/Architect adjusts taxonomy using drag-drop UI

5. New versions are saved and tracked in version history

6. Users can export or import taxonomies in JSON or CSV

7. System supports rollback and change comparison across versions

1. **AI HCM Product 7: AI-Assisted Goal & Feedback Writing**

* 📛 Name Suggestion (Product-Level):

**GoalGenie**  
(Alternatives: SMARTWriter, FeedbackFlowAI, AimCraft, ReflectAI)

* 🎯 Goal / Purpose:

To assist employees and managers in writing clear, effective goals and performance feedback using AI-driven suggestions, aligning with the SMART framework, while streamlining review cycles and improving communication quality.

* 📦 Product Description:

**GoalGenie** is an AI-powered solution that simplifies the creation of work goals and feedback across quarterly or yearly review cycles. It integrates generative AI to draft SMART-aligned goals and uses sentiment analysis to evaluate and guide feedback tone. Users receive real-time suggestions while inputting content, and all historical goals and evaluations are accessible and contextually reused to ensure alignment and continuity.

* 🏗️ System Architecture:

User Input → AI Suggestion Engine (Goal Writer + Feedback Analyzer)

↓

SMART Compliance Checker + Sentiment Scorer

↓

Review + Edit UI → Save to Evaluation Database

↓

History Retrieval API + Analytics Dashboard

* 🖥️ UI Components (Frontend):

| **Component** | **Description** |
| --- | --- |
| <GoalInputWithAIHint /> | Text input field with AI-generated SMART goal suggestions |
| <SMARTGuidancePanel /> | Visual cues to ensure goals are Specific, Measurable, Achievable, Relevant, Time-bound |
| <FeedbackEditor /> | Feedback entry with tone/sentiment score and rewrite suggestions |
| <HistorySidebar /> | View previous cycles’ goals and feedback for context |
| <ReviewPeriodSelector /> | Toggle between different review quarters or years |
| <SubmitAndSaveModal /> | Final step to validate and confirm entries |

* 🔐 Access Rights:

| **Role** | **Permissions** |
| --- | --- |
| Employee | Create/edit own goals and feedback; view history |
| Manager | Review, comment, and adjust direct reports’ goals/feedback |
| HR Reviewer | Full access to all submissions, analytics, and exports |
| Admin | Configuration of review periods, AI parameters, and access policies |

* 🧠 AI Models and Recommendation Logic

📥 **Input:**

* Draft text for goals and feedback
* Historical performance records (optional)
* Role, function, and KPIs from system context

📤 Output:

* SMART-optimized goal suggestions
* Rewritten goals if input is vague or poorly structured
* Feedback tone classification (positive/neutral/constructive/negative)
* Suggested improvements to tone and clarity

#### 🧠 AI Models Used**:**

* **Goal Generation Model:** Fine-tuned LLM (e.g., GPT) using SMART goal structure
* **Sentiment Analysis Engine:** Text classification model (e.g., BERT-based) for feedback tone
* **Rewriting Assistant:** Prompt-based GenAI for clearer, more actionable statements
* **SMART Validator:** Rule-based checker to ensure goals follow SMART format
* 🧰 Backend Architecture

📡 API Layer:

| **Endpoint** | **Description** |
| --- | --- |
| POST /ai/generate-goal | Returns SMART-aligned goal suggestion |
| POST /ai/analyze-feedback | Classifies tone and gives suggestions |
| GET /goals/history | Retrieves goals by quarter/year/user |
| POST /goals/save | Saves or updates goal/feedback entry |
| GET /review-periods | Fetches list of available cycles |

🔧 Services:

* **Goal & Feedback Generator Service**
* **Sentiment Analysis Service**
* **Evaluation History Service**
* **SMART Validator Engine**
* **Access Control & Role Management**
* 🛠️ Suggested Stack

| **Layer** | **Technology** |
| --- | --- |
| Frontend UI | React, Tailwind CSS, Quill.js or TipTap (for rich text editor) |
| Backend API | Node.js or FastAPI |
| AI/ML Models | OpenAI GPT / LLaMA 2 (prompt-tuned), BERT (sentiment analysis) |
| Database | PostgreSQL (goal records), Redis (caching), MongoDB (drafts) |
| File/Versioning | S3 or Blob Storage for attachments and audit trails |
| Auth & Identity | OAuth2 / SSO (Azure AD, Google, etc.) |
| Deployment | Docker + Kubernetes, with monitoring/logging (ELK or Grafana stack) |

* 🔄 Data Flow Summary

1. User accesses goal/feedback form for a review cycle

2. Inputs draft → GenAI suggests improvements

3. SMART framework checks for quality assurance

4. Feedback tone is analyzed → suggestions provided

5. User saves submission → stored per user, period, and manager

6. Manager/HRBP can review and export data or access historical context

1. **AI HCM Product 8: Smart Learning Path Recommendations**

* 📛 Name Suggestion (Product-Level):

**PathWise**  
(Alternatives: LearnSphere, SkillRouteAI, EduNavi, GrowTrack)

* 🎯 Goal / Purpose:

To provide personalized, AI-driven learning paths that recommend relevant internal and external courses aligned with an employee’s current role, career aspirations, and skill gaps.

* 📦 Product Description:

**PathWise** is a smart learning recommendation system that maps employee roles and skill needs to curated internal and external courses (LMS, Coursera, LinkedIn Learning, etc.). The AI engine recommends paths based on current and desired roles, enables personalized goals, and tracks progress. It empowers employees to own their development while giving HR visibility into upskilling and reskilling across the organization.

* 🏗️ System Architecture:

User Profile + Career Goal → Role & Skill Mapping

↓

AI Recommendation Engine

↓

Learning Dashboard (with course suggestions, goals)

↓

LMS API Sync (track enrollment, progress, feedback)

↓

HR Analytics Dashboard + Data Warehouse

* 🖥️ UI Components (Frontend):

| **Component** | **Description** |
| --- | --- |
| <LearningDashboard /> | Personalized dashboard with course cards, goals, and recommendations |
| <SetLearningGoalModal /> | UI for employees to define specific learning goals (e.g., target role, skill) |
| <RecommendedCoursesList /> | Display courses (internal, LinkedIn, Coursera) |
| <SkillToCourseMap /> | Visual display showing how each course addresses relevant skill gaps |
| <ProgressTracker /> | Monitors ongoing course progress and completion rate |
| <FeedbackInput /> | Allows users to rate and comment on completed courses |

* 🔐 Access Rights:

| **Role** | **Permissions** |
| --- | --- |
| Employee | View personalized dashboard, set goals, enroll and track progress |
| Manager | Recommend learning goals to team members, view progress |
| HR/LD Team | Configure taxonomies, manage course libraries, track learning across org |
| Admin | Manage integrations, user access, and AI rules/pipelines |

* 🧠 AI Models and Recommendation Logic

📥 **Input:**

* Current role and competencies
* Career aspiration / target role (if defined)
* Skill gaps (inferred or assessed)
* Internal course catalog
* External course metadata (via APIs from LMS, LinkedIn, Coursera, etc.)

📤 Output:

* Ranked list of recommended courses
* Suggested learning paths aligned with goals
* Skill coverage visualization
* Learning progress and feedback analytics

🧠 AI Models Used:

* **Recommendation Engine:** Content-based + collaborative filtering (hybrid)
* **Embedding Model:** Vector similarity between skills and course descriptions (e.g., Sentence-BERT)
* **Path Optimizer:** Graph-based role-skill-course mapping
* **Feedback Loop Trainer:** Reinforcement learning based on user behavior and feedback
* 🧰 Backend Architecture

📡 API Layer:

| **Endpoint** | **Description** |
| --- | --- |
| POST /ai/recommend-courses | Returns personalized course suggestions |
| GET /user/learning-dashboard | Fetch learning goals, progress, and course recommendations |
| POST /goal/set | Save user's learning objectives |
| POST /course/feedback | Submit ratings and reviews |
| GET /external-courses | Pull courses from external LMSs via API |
| GET /skills-mapping | View skill ↔ course relationships |

🔧 Services:

* **Learning Path Recommendation Service**
* **Skill-Course Mapping Service**
* **LMS Integration Service (Coursera, LinkedIn, etc.)**
* **Goal Tracking Service**
* **Feedback Collection and Relevance Adjustment Service**
* 🛠️ Suggested Stack

| **Layer** | **Technology** |
| --- | --- |
| Frontend UI | React, Tailwind CSS, D3.js (for skill-course maps) |
| Backend API | FastAPI or Node.js |
| AI/ML Models | Sentence-BERT, FAISS (vector search), XGBoost (ranking), Neo4j (role-skill graph) |
| Database | PostgreSQL (user data), MongoDB (course metadata), Redis (caching) |
| LMS Integration | OAuth2 + REST APIs for LinkedIn Learning, Coursera, Udemy |
| Auth & Identity | SSO (Azure AD, Google, etc.) |
| Deployment | Docker + Kubernetes, Prometheus + Grafana for monitoring |

* 🔄 Data Flow Summary

1. User defines learning goal (e.g., target role, skills to acquire)

2. AI maps current role & skills → target profile → required learning path

3. Recommender fetches course matches (internal/external APIs)

4. UI displays dashboard with course suggestions, skill alignment, and progress tracking

5. LMS APIs track enrollment and completion

6. Feedback is collected to improve recommendations

7. HR and managers access insights through analytics dashboard